양자컴퓨팅 소프트웨어 최신 기술 동향

State-of-the-art in Quantum Computing Software

조은영 (E.Y. Cho, eycho@etri.re.kr) 클라우드기반SW연구실 책임연구원
김영철 (Y.C. Kim, kimyc@etri.re.kr) 데이터중심컴퓨팅시스템연구실 책임연구원
정희범 (H.B. Jung, hbjung@etri.re.kr) 클라우드기반SW연구실 책임연구원
차규일 (G.I. Cha, gicha@etri.re.kr) 클라우드기반SW연구실 책임연구원

ABSTRACT

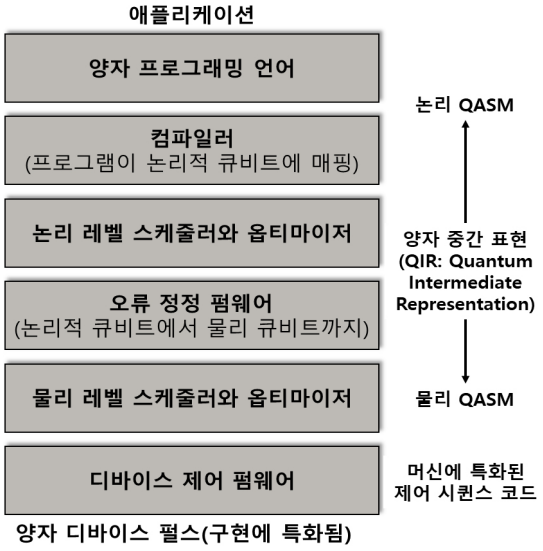
Since Richard Feynman presented the concept of quantum computers, quantum computing have been identified today overcoming the limits of supercomputing in various applications. Quantum hardware has steadily developed into 50 to hundreds of qubits of various quantum hardware technologies based on superconductors, semiconductors, and trapped ions over 40 years. However, it is possible to use a NISQ (Noisy Intermediate Scale Quantum) level quantum device that currently has hardware constraints. In addition, the software environment in which quantum algorithms for problem solving in various applications can be executed is pursuing research with quantum computing software such as programming language, compiler, control, testing and verification. The development of quantum software is essential amid intensifying technological competition for the commercialization of quantum computers. Therefore, this paper introduces the trends of the latest technology, focusing on quantum computing software platforms, and examines important software component technologies.

KEYWORDS quantum compiler, quantum computing, quantum computing software

Ⅰ. 서론

양자컴퓨팅 소프트웨어 계층에서 그림 1과 같이 양자 프로그래밍 언어로 기술된 양자 알고리즘은 컴파일러를 통해 특정 양자 하드웨어 도메인에 최적화된 양자 명령어로 변환된다[1,2]. 이 과정에서 현재 양자컴퓨팅 구현 수준에 따라 양자 오류 정정 과정이 추가되고, 논리 큐비트 연산이 다수의 물리 큐비트에 기반한 실제 물리 연산으로 변환된 후 최종 단계에서 대상 양자 프로세서를 구동하는 제어 펄스를 생성하는 양자 명령어로 변환된다. 따라서 양자컴퓨팅 환경은 고전컴퓨팅과 유사한 정도의

* DOI: https://doi.org/10.22648/ETRI.2021.J.360607
* 본 연구 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임[No. 2020-0-00014, 결합 허용 논리양자큐비트 환경을 제공하는 양자운영체제 원천기술 개발].



출처 Reprinted with permission from [1].

그림 1 일반적인 양자 프로그래밍 도구흐름

추상화 수준과 컴파일 및 디버깅 도구를 갖추어야 더 효율적이고 프로그래밍 오류가 최소화된 양자 알고리즘의 구현이 가능하게 된다.

최근 다양한 양자 소자 기술에 기반한 양자 하드웨어 기술 연구가 빠르게 진척되어 50~150큐비트 규모의 양자 프로세서 개발로 이어지고 있다. 그러나 이 양자 하드웨어는 아직 독자적인 컴퓨팅 시스템으로 운영되기에는 구성 요소의 기술 완성도가 낮아 고전 컴퓨터의 이종 컴퓨팅 환경과 같은 계산 가속기 형태로 운영되고 있다[3]. 즉, 고성능 컴퓨터의 계산 가속기 형태로 양자컴퓨터를 연동하여 이종 컴퓨팅 환경으로 활용한다. 이와 같은 계산 가속기 유형의 상용 양자컴퓨팅 시스템으로는 Qiskit(IBM), Cirq(Google Quantum AI), QCS(Rigetti), Azure Quantum(Microsoft), Braket(AWS), Quantum Leaf(Baidu), Alibaba Quantum Computing Cloud(Alibaba)를 제공하고 있다[4,5]. 또한 초전도체 방식의 Forest(Rigetti), 이온 트랩 방식의 IonQ, 양자 어닐링 방식의 Ocean(D-Wave) 플랫폼 등이 있다. 현재 다양한 양자 큐비트 소자 기술이 경쟁하고 있지만, 본고는 초전도체 기반 최근 3년간의 양자컴퓨팅 플랫폼을 중심으로 양자 프로그래밍 언어와 컴파일러, 시스템 검증 기술 관련 분야에 국한하여 최신 기술 동향을 정리하였다[6-10]. 최종적으로 본고에서는 양자컴퓨팅 소프트웨어 계층의 요소 기술 동향을 분석하고, 이 분야에서 향후 해결해 가야 할 기술적 도전 과제에 대해 분석하고자 한다.

II. 양자컴퓨팅 소프트웨어 플랫폼 동향

최근 양자 기술 투자와 성과들이 활발하여 국내외에서 주요 양자 플랫폼 관련 연구가 활발히 이루어졌다[11-13]. 대규모 선도 기업을 포함한 전반적인 상황은 표 1의 요약을 참조하고 본고에서는 특히 진보된 연구가 이루어진 Forest, Qiskit, ProjectQ, ScaffCC, XACC, t|ket〉, OpenQL 등의 플랫폼 내 큐비트 배치, 스케줄링, 최적화를 수행하는 컴파일러 특징과 동향을 중심으로 소개한다[14,15].

1. Forest

Rigetti Computing에서 개발한 Forest 플랫폼은 Quil이라는 어셈블리 언어를 제공하고, pyQuil이라는 고급 언어로 프로그래밍에 접근한다. 현재는 Aspen 128큐비트 규모의 성능이 공개되어 있고 이 장치와 연동된 플랫폼들도 다수 존재한다. Forest는 위상을 반영한 컴파일러를 제공하고 잡음 반영 모의시험도 진행되었다[16]. 두 단계로 구성된 컴파일러 QUILC는 경험적 기법을 사용한 논리 큐비트 주소 부여 단계(Addressing Stage)와 국소적 최적화

표 1 주요 양자 소프트웨어 플랫폼 요약

Platform	Forest	Qiskit	ProjectQ	QDK	Cirq	ScaffCC	XACC	t|ket>	OpenQL
Institution	Rigetti	IBM	ETH Zurich	Microsoft	Google Quantum AI	Princeton	ORNL	CQC	QuTech
First Release	v0.0.2 on Jan 15, 2017	0.1 on March 7, 2017	v0.1.0 on Jan 3, 2017	0.1.1712.901 on Jan 4,2018	v0.1 on Jul 1, 2018	v1.0 on Aug 6, 2016	-	-	v0.5.3 on Oct 11, 2018
Version	v3.0 on Aug 3, 2021	0.29.0 on Aug 3, 2021	v0.7 on Aug 3, 2021	0.18.2106.148911 on Jun 25, 2021	v0.11.1 on Aug 3, 2021	v5.0 on Apr 23, 2020	- (on Jun 30, 2020)	v0.3 on Aug 15, 2021	v0.10 on Aug 3, 2021
Open Source	✓	✓	✓	✓	✓	✓	-	✓	✓
License	Apache-2.0	Apache-2.0	Apache-2.0	MIT	Apache-2.0	BSD-2	BSD-3	Apache-2.0	Apache-2.0 (O) GLPK
OS	Mac, Windows, Linux	Mac, Windows, Linux	Mac, Windows, Linux	Mac, Windows, Linux	Mac, Windows, Linux	Mac, Linux	Mac, Linux	Mac, Windows, Linux	Mac, Windows, Linux
Requirements	Python 3, Anaconda	Python 3.5+, JupyterNotebooks Anaconda 3	Python 2 or 3	Visual Studio Code	Python 3.5+, Anaconda 3	Python 3, Anaconda LLVM 10	Python 3.7, Anaconda	Python 3.7, JupyterNotebooks, Anaconda	Python 3, Anaconda
Classical Host Language	Python	Python, C++ JavaScript, Swift	Python, C++	C#	Python	C/C++	C++ (Python, Julia)	Python, C++	Python, C++
Quantum Prog. Language	pyQuil	Qiskit	ProjectQ Silq	Q#	-	Scaffold, Scaffold-NISQ	QCOR	Qiskit, Cirq, pyQuil, ProjectQ, PyZX	-
Quantum Language	Quil	OpenQASM	-	-	-	OpenQASM	XASM OpenQASM Quil	IBM-Q, Aer, Rigetti QCS, QVM, ProjectQ Sim	cQASM eQASM
Quantum Hardware	Aspen-9(31 qubits) (128 qubits – '20)	IBMQX2,4 (5 qubits), IBMQX5 (16 qubits), QS1 1 (20 qubits) 65 qubits	no dedicated hardware, connect to IBM backends	-	Foxtail (22 qubits) Sycamore (54 qubits) Bristlecone (72 qubits)	-	-	-	17 qubits (49 qubits)
Simulator (Benchmark)	~20 qubits locally, 26 qubits with most APIkeys to QVM, 30+ w/private access	~25 qubits locally, 50 through cloud Qiskit Aer	~28 qubits locally	30 qubits locally, 40 through Azure cloud	20 ~ 30 qubits Simulator XmonSimulator	128 qubits QX	5 ~ 85 qubits Eclipse TNQVM 65 qubits IBM	~ 26 qubits 53(IBM Rochester) 53(Google Sycamore) 16(Rigetti Aspen)	26 qubits QX, (28 qubits QBeeSim)
Features (Compiler)	-대규모 신속 모의시험 JIT 컴파일 모드, Quil Interpreter - 위상 반영 컴파일 - 잡음반영 모의시험 - 상태 인식 (부분) 컴파일 - addressing/compression - peephole rewriter, retargetable - tweedledum 외부Lib. - SWAP선택: A*heuristic	- QASM 생성 - 위상/잡음 반영 컴파일러 - 회로 편집기 - Aqua/Terra 라이브러리 - Qiskit Slack 소통채널 - ~ 30 compiler passes - Terra compiler 4단계 (7개 transpiler ~ LLVM) - Z3 map SMT solver	- 회로 편집기 -다수라이브러리 Plugin (Fermion, Mathlib) - 큐빗 배치와 자원 사용 계산 - HW제약없음 (IBM, IonQ, AWS)	-화학, 기계학습, 수치 라이브러리 -1Qbit, IonQ, Honeywell 협력 - azure quantum solver로 가속화 - 상용QIO solver - 회로의 동시성 길이/폭 자원추정 - 다양한 FE/BE 연동용 공통 LLVM QIR	- 매개변수화 회로 최적화 처리 - OpenFermion, qsim, TensorFlow, PennyLane 플러그인 연동 - HW제약없음 - 2/3/4차원 Qid 정의 (qubit/qutrit/ququart) - 사용자 병렬 moment 반영 스케줄링 - 21개 최적화 모듈 - pure/mixed state 및 외부 고성능 모의시험	-확장성(trillion급) - QASM-H/HL - LLVM 기반 컴파일과 분석 -CTQG(Classical- To-Quantum-Gate) -자원 계수기 - Multi-SIMD 병렬 -잡음 적응형 - qjit(just-in-time) -OpenQASM연동 - 경량 Scaffold (50~100 qubits) - TriQ: Z3 SMT	- 모듈화 - 확장성 - SOA (service- oriented SW archi.) - 교차 플랫폼 - LLVM/Clang 기반 - jit_compile 제공 -IBM,Rigetti,D-Wave -응용: 에너지 - SABRE 적용	- 언어/장치 제약 없는 컴파일러 - 회로 재작성 엔진 -핍홀/거시적 최적화 - NAGP: 잡음 인식 그래프 배치 방법 - Hoare Triple - 임의 각도 - 부분 컴파일 - 응용: 화학(Eumen), 자연어 처리	- 융통성 - 이식성, - 회로 편집기 - FT QEC 로드맵 - 초전도/스핀 큐빗 - 컴파일 패스 구성 -Floyd-Warshall Algo. (distance matrix 구축) - 응용: 얽기서열 금융 - 마이크로아키텍처 SIMD 및 축약된 MIMD 병렬 연산

를 위해 peephole rewriter가 포함된 압축 단계(Com-pression Stage)로 이루어진다. 실행 전 상태에 따라 등가성 회로 대체가 달라질 수 있으므로 상태 인식(State-aware) 컴파일 기법도 반영되어 있다. 즉, 부분 컴파일을 허용하여 데이터 가용성 정보를 인지한 큐비트 자원 일정 계획이 가능하게 된다. 또한 Quil Interpreter로 알고리즘 작성을 개선할 수 있는 도구가 잘 제공되어 있다.

Tweedledom이라는 양자 회로 합성, 조작 및 최적화용 외부 컴파일 라이브러리로 높은 자동 재사용성을 제공한다. 두 큐비트 동작 시 인접성 요구사항에 의한 큐비트 교환(SWAP) 선택 방법은 A* 휴리스틱 알고리즘으로 제공하고 있다. 성능 정보는 무작위 벤치마킹과 양자 오류 수정을 위한 안정기(Stabilizer) 모의시험 진행상황이 공개되어 양자 오류 정정 분야에서도 좋은 참조가 된다.

2. Qiskit

IBM QX(Quantum Experience) 양자 클라우드에서는 이미 171개국에서 10만 명 이상의 사용자를 확보하고 2019년 기준 530만 건 이상의 양자 디바이스 실험과 1,200만 건 이상의 가상시험을 수용하고 있다[17]. IBM은 타 기관 플랫폼과의 연동이 활발하게 이루어져 성능 시험 정보가 많은 사례이다. 매일 실제 하드웨어의 오류 성능 정보도 갱신하여 이를 반영한 큐비트 사용 알고리즘을 작성할 수 있도록 제공하고 연동된 플랫폼들에서 활용하는 상황이다. 양자컴퓨터 프로그래밍 실습이 가능한 Qiskit Composer는 Drag-and-drop 방식으로 손쉽게 회로를 작성하고 양자 상태 벡터를 블로흐(Bloch) 구면에 표현하는 동기화된 가시화를 제공한다.

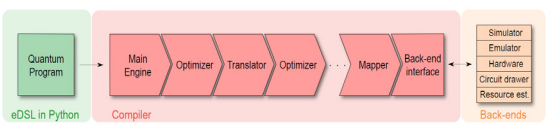
중국과 기술 우위 경쟁을 하는 가운데 모의시험은 Qiskit Aer상에서 50큐비트 이상의 시험이 진행되었다. 위상/잡음 반영 컴파일러와 Aqua 및 Terra 라이브러리가 제공되고 벤치마크 데이터가 풍부하며, 특히 slack 채널을 통해 다양한 문제 해결과 자료 공유도 활발하여 최신 정보를 참조한다.

3. ProjectQ

2016년에 발표된 취리히 공과대학교의 양자 플랫폼은 슈퍼컴퓨터 수준의 고전 컴퓨터에 의해 제어되는 양자 가속기 모델로 설계되었다.

그림 2와 같이 양자컴퓨팅 도구 체인에서는 기존 인프라 활용을 위한 임베디드 언어로부터 시작하여 시뮬레이터, 에뮬레이터, 자원 분석기, 하드웨어 등의 백엔드를 고려하였다[18]. 계층화된 방법론으로 모듈화와 유연성을 제공하는데 하드웨어에 구애되지 않는 상위 수준 컴파일러의 최적화를 제공한다. 오류 보정(QEC: Quantum Error Correction) 단계를 전후로 QIR(Quantum Intermediate Representation)과 LLQIR(Low-Level QIR)을 생성하고 이런 계층화를 통해 독립적 구현과 그로 인한 코드 재사용률이 증가한다. 자원 계수기는 알고리즘 실행 시 사용할 큐비트와 게이트 수를 추적하여 알고리즘 개선 최적화를 달성한다.

한편 라이브러리에 관한 최적화 연구도 진행된 바 있는데 양자컴퓨팅의 공간 제약이 훨씬 심하므로 추가 메모리의 부담이 적은 in-place 알고리즘



출처 Reprinted from [18], CC-BY 4.0.

그림 2 ProjectQ 전 과정 프레임워크

이 유리하지만 계산되지 않은 중간 결과들로 인해 시간이 더 많이 소요될 수 있다. 따라서 비계산 중간 결과와 더 많은 큐비트 수요 간의 절충점을 찾는 것이 중요하다. 보조 큐비트 레지스터를 초기화하는 일은 다시 계산되어야 하는 부분을 포함하는데 2020년에 배포된 연구에 따르면 대형 함수에 대해 공간 복잡성을 25% 수준으로 줄이는 방식을 적용하여 해결하였다[19].

4. ScaffCC

ScaffCC는 LLVM(Low Level Virtual Machine) 기반 확장형 컴파일과 분석 프레임워크로 미국 프린스턴 대학교에서 개발하였다. 고전 프로그래밍 환경에서 친숙한 C 언어를 확장하여("put things together" 한곳으로 모으는 의미로 생성된 양자 언어인) Scaffold로 표현된 회로 알고리즘을 QASM(Quantum ASseMbly language)으로 생성하는데, 양자 프로그램 정확도 검사와 데이터 흐름 추적 등이 중요하므로 특별히 분석 단계에 중점을 두고 개발되었다[20,21]. LLVM IR을 통해 최적화 변환 단계를 유용하게 사용하는데 컴파일 이후에 다시 QASM-to-IR 변환으로 복제 불가, 얽힘, 자원, 모듈화, 시간 분석 등을 수행하고 있다.

수조(trillion) 개 수준의 명령이 포함된 프로그램까지 처리하도록 설계되어 확장성 측면에서 좋은 참조 사례이다. ScaffCC에서의 특이한 고려사항은 CTQG(Classical-To-Quantum-Gate Conversion) 모듈 컴파일 개념이다. 양자 알고리즘에서 모듈의 많은 부분이 고전적인 가역 논리 연산을 사용하는 특성에 따라 이러한 classical oracle 부분은 flat QASM으로 변환하여 최종 호출 부분에 그대로 추가하여 개발자가 고전컴퓨팅 환경에서 검증/디버깅을 수행하며 불필요한 양자 비트/게이트로 변환하는 오버

헤드와 성능 저하를 방지할 수 있게 된다. 컴파일러가 높은 수준의 timing 분석을 진행하게 되면 회로 최적화를 위한 명령어 재정렬로 임계 경로 길이를 추정한다.

양자 프로그램의 추적은 용량이 매우 커질 수 있으므로 모듈화를 활용하게 되는데, 주요 경로를 찾아낸 뒤 모듈 간에 존재할 수 있는 병렬성을 위해서는 평탄화를 진행하여 개선하게 된다. 이러한 방식은 fine-grained scheduling 방식에 속하게 되고 모듈 내 피연산자들의 임계 경로 차이로 인해 벌어진 부분을 다시 최대한 좁히는 slack-aware critical path scheduling으로 추가 최적화를 도모하고 있다. 양자 얽힘(Entanglement)은 큐비트 상태의 최종 결과에 영향을 미치는 현상이므로 알고리즘 설계와 디버깅에 유효한데 분석 단계에서 이를 추적할 수 있는 기능이 제공된다. 또한 보조 큐비트의 상태 복귀를 위한 uncompute와 disentanglement 식별, 추적, 예상 시간 추정, 큐빗 이동 인지 Multi-SIMD(Single Instruction, Multiple Data) 병렬 구조 등의 연구 기법으로 많은 참조가 이루어졌다.

5. XACC

XACC(eXtreme-scale Accelerator Programming Framework)는 미국 에너지부의 지원으로 오크리지 국립 연구소에서 개발한 시스템이다. 양자-고전 컴퓨팅 작업의 일부를 호스트 CPU에서 연결된 양자 가속기로 보내어 결과를 계산하고 요청한 시스템으로 다시 보내는 방식의 가속화된 노드 모델로 양자-고전 상호 작용을 최적화하도록 적용하였다[22]. 자체 개발 장치도 수용하고 IBM, Rigetti, D-Wave 및 IonQ 플랫폼과도 함께 작동하여 하드웨어에 제약되지 않는 프레임워크를 구축하였으며 온라인 제공시스템도 지원할 예정이다. 상호운용성을 핵심으로 게이트 기반과 어닐링이라는 양자컴퓨팅 모델을 수용하며 단일 공통 언어로 다양한 양자컴퓨팅 플랫폼과 응용을 구성하고 컴파일하는 교차 플랫폼으로 작동한다.

XACC는 3계층 구조로 이루어져 먼저 front-end 양자 표현 코드가 middle-end의 XACC IR로 변환되고 이 인스턴스가 back-end 계층의 하드웨어 장치에서 실행하는 추상 양자 인터페이스로 확장 제공된다. 이때 M개의 양자 장치에 N개의 언어를 제공하기 위해 M*N 개가 아닌 M+N 개의 대응만 필요하게 되므로 작업량을 대폭 줄이게 된다. 이는 올해 인텔이 LLVM 기반으로 C/C++ 컴파일러를 배포한 이점과도 같다[23]. XACC에 정의된 가속기, 명령어, IR 변환, 컴파일러, 알고리즘 등에 대해 서비스 인터페이스를 제공할 수 있도록 하는 SOA(Service-Oriented Architecture)를 제공하여 모듈 확장성을 제공한다.

양자-고전컴퓨팅 자동화 목적의 QCOR은 XACC 기반 Software Stack으로써 가속기 버퍼 개념을 활용하여 호스트 컴퓨터와 양자 제어시스템 모두가 접근할 수 있는 공유 메모리 모델 및 비동기 실행 모델을 사용한다[24]. 이 컴파일러는 양자 회로 최적화와 스케줄러, 실행 시간 오류 완화 등을 제공한다. 실행 시간 모델에서는 XACC accelerator class 확장을 통해 NISQ(Noisy Intermediate-Scale Quantum, 잡음이 있는 중간 규모 양자) 모델에서의 양자-고전 통합 효용성을 촉진하고 FTQC(Fault-Tolerant Quantum Computing, 오류 정정이 이루어진 양자) 모델도 제공할 수 있도록 제시되어 있다. 패턴 매칭 기반 회로 최적화, 단일/쌍 큐비트 등가 및 회전 결합 최적화 패스를 포함하고 하드웨어 배치 전략도 현재까지 최선의 알고리즘으로 알려진 UC 산타바바라의 SABRE(SWAP-based BidiRectional heuristic search algorithm)까지 반영된 상태이다.

6. t|ket>

최근에 공개되고 발전된 영국 캠브리지 양자 센터의 retargetable compiler인 t|ket〉은 LLVM 유형을 따라 구현되어 양자 언어에 구애되지 않고 다양한 하드웨어에 맞춘 교차 컴파일러 역할을 할 수 있는 플랫폼이다[25]. NISQ 하드웨어에 대한 코드를 생성하도록 설계되고 컴파일러의 최적화 패스 조합에 대한 정확성 보장을 위해 Hoare Triple 개념을 적용하고 있다. 회로 최적화는 peephole과 macroscopic optimization을 제공하고 있으며 NP-complete 문제인 물리 큐비트 초기 배치에는 Noise aware graph placement 기법을 적용하여 해결하고 있다. Qiskit, Cirq, PyQuil, ProjectQ, PyZX 등 다양한 많은 기존 플랫폼과 연동되어 성능 자료를 포함한 주목할만한 사례이다.

7. OpenQL

네덜란드 QuTech와 델프트 공대에서는 상위 수준의 양자 프로그래밍 언어와 관련 양자 컴파일러를 포함하는 OpenQL이라는 양자 프로그래밍 프레임워크를 구현하였다[26]. OpenQL은 초전도와 실리콘 기반 스핀 큐비트 등 두 개의 서로 다른 큐비트 기술에서 동일한 고수준 알고리즘을 실행하였다. OpenQL은 또한 실행 코드 외에도 하드웨어 기술과 독립적이며 QX 시뮬레이터를 사용하여 모의시험할 수 있는 중간 양자 조립 코드인 cQASM (common QASM)도 생성한다. Software full stack 구성으로 장치 독립적인 계층과 종속적인 계층으로 나누어 큐비트 기술별로 장비를 제어하는 마이크로 명령어 구조까지의 연동이 이루어져 실질적인 연구가 활발하게 이루어지고 있는 사례이다. 명령어 묶음이나 SIMD(Single Instruction Multi Data) 및 축약 MIMD 연산 형태로 병렬성을 지정하여 처리할 수 있고 두 개의 서로 다른 장치에 대해서 하드웨어 설정 파일을 활용하여 동일한 상위 OpenQL 코드를 사용할 수 있도록 함으로써 이식성과 융통성이라는 설계 목표를 실현하였다. Surface-17이라는 NISQ 장치를 대상으로 무작위 벤치마킹(RB), AllXY 등의 시험이 이루어졌으며 2021년도에는 창업기업 QBeeX에서 QBeeSim이라는 슈퍼컴퓨터 기반 모의시험으로 유전체 염기서열 분석과 금융 분야에서의 효과적인 양자 알고리즘 개발에 지속 기여하고 있다[27].

III. 양자 프로그래밍 환경 동향

확장성이 있는 양자컴퓨터의 소프트웨어 구성 요소는 양자 알고리즘을 표현하는 프로그래밍 언어와 컴파일러, 양자 명령어 정의 구조(QISA: Quantum Instruction Set Architecture), 시뮬레이션, 검증과 디버깅 등의 영역이다. 고전 컴퓨터 CPU에 대한 turing machine 개념 모델보다는 QRAM(Quantum Random-Access Machine) 개념 이후 최초의 실용적인 양자 언어 QCL이 출현하였고 이후 오늘날까지 꾸준히 함수형 혹은 명령형 언어 형식으로 다양하게 제안되었다.

표 2에 정리한 양자 언어 구조 연대기를 보면 2016년부터 3년간 가장 활발하게 발표되었고 지금도 양자 알고리즘의 특성을 반영하고 사용성을 개선하기 위한 연구가 진행되고 있다[28,29]. 밑줄로 표시된 양자 어셈블리 언어(QASM: Quantum ASeMbly language)들 중에는 OpenQASM의 영향력으로 실제적인 표준 위치에 있고 가장 최근에 이온트랩 방식을 위한 Jaqal이라는 언어가 발표되고 현장에서의 교육 등을 진행하고 있다[30].

최근의 고급 언어로써 취리히공대의 Silq라는 언

표 2 양자 언어 구조 연대기

Year	Language	Year	Language
1996	Quantum Pseudocode	2016	ProjectQ FJQuantum pyQuil, Quil QASM QMASM
1998	QCL		
2000	qGCL		
2003	Lambda Calculus Q language		
2004	QFC, QPL CQP	2017	Forest, Qiskit IQu, QWIRE Proto–Quipper–M OpenQASM2 cQASM/eQASM
2005	QML QPAlg cQPL		
2006	LanQ		
2008	NDQJava	2018	Strawberry Fields Blackbird QuantumOptics.jl Cirq, Q|SI⟩, qPCF, Q#, OpenPulse
2011	QuECT		
2012	Scaffold		
2013	QuaFL Quipper Chisel–Q	2019	XASM
		2020	Silq, QUA, Jaqal, QUASAR, qV
2014	LIQUi|⟩		
2015	Proto–Quipper	2021	OpenQASM3 FunQ, Quala

출처 Reproduced from [28,29].

어는 보조 큐비트를 계산 이전 상태로 되돌리는 절차를 자동화하거나 사용자 오류를 예방하는 절차를 제공하고 있다[19]. 이처럼 유용한 양자컴퓨터를 위해서 하드웨어, 알고리즘의 개발과 동시에 소프트웨어 도구의 연구개발이 함께 발전해 가야 한다. 양자 프로그래밍과 컴파일은 알고리즘이 갖는 추상적인 수학적 내용을 실제 컴퓨터에서 실행할 수 있도록 변환하는 중요한 작업이다. 프로그래밍 언어는 핵심 개념과 동작을 지원하는 표현 문법을 제공해 변환 절차를 지원하게 되는 것이다.

최상위 수준의 언어는 알고리즘을 쉽고 빠르게 프로그래밍할 수 있고 장치 독립적이며 이식성이 있는 소프트웨어를 많이 생성할 수 있다. 또한 하위 수준의 언어는 하드웨어 구성 요소와 원활한 상호 작용으로 신속한 실행이 되도록 제공하게 된다. 수십억 개의 트랜지스터로 이루어진 고전 컴퓨터에서 프로그래머는 그 오류에 대해 전혀 신경 쓰지 않고 자유로이 구현하고 디버깅하는 것처럼 장기적으로는 양자컴퓨팅도 자동화된 도구 흐름에서 하위 수준 언어를 흡수하여 추상화를 제공하게 되어야 할 것이다.

상위 수준의 양자 언어는 함수형과 명령형이 모두 개발되었고 아직 그 적합성의 결론이 나지 않은 상태이다. Q#, Quipper, QuaFL, LiQui|⟩ 같은 함수 언어는 추상적이거나 수학적인 알고리즘 구현에 적합하여 간결하고 오류가 적은 코드가 가능하다. Scaffold, ProjectQ 같은 명령형 언어는 변수의 직접적인 수정과 자원 효율적인 설계를 지원한다. 또한 공식적으로 정의된 언어를 확장하는 embedded 언어는 개발자가 기존 언어 환경을 활용하여 초기 구현 속도를 높일 수도 있다.

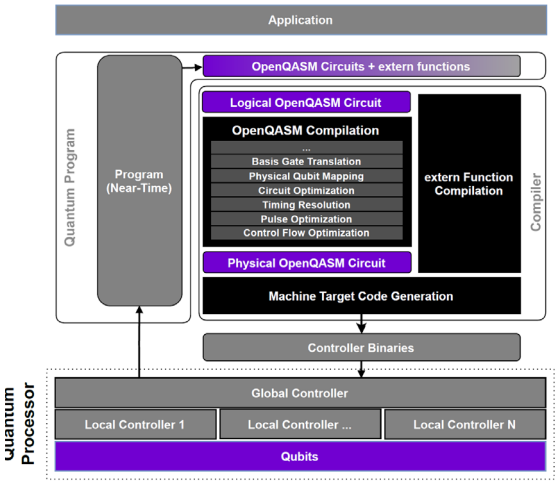
본고는 비교적 초기 양자 언어이지만 확장성이 잘 고려되고 최근까지 활발한 개선과 연동이 이루어지고 있는 Scaffold 언어에서 고려한 사항들을 살펴보면서 양자 언어와 컴파일러에서 고려하는 사항을 함께 확인하고자 한다[20].

고급 언어로 표현된 알고리즘은 컴파일러에 의해 기계어로 변환되는데 루프를 단순히 펼치게 되면 확장성에 문제가 된다. 이에 모듈화를 하여 출력 파일의 크기를 줄임으로써 분석 시간과 메모리 공간을 개선할 수 있도록 다양한 선택을 제공한다. 또한 QASM–HL(with Hierarchy and Loop) 방식으로도 정적 분석 단계(Pass–Driven Approach)에 중간 표현이 크게 생성되는 경우는 실행 시간에 변환하는 방식(Instrumentation–Driven Approach)을 제공하여 출력 코드의 크기를 억제하기도 한다. 즉, 고전적 의존성은 고전적으로 해결하고 양자 부분의 정보를 수집함으로써 알고리즘의 특성에 따라 적용할 수 있도록 제공되는 것이다. Silq와 마찬가지로 회로의 가역적(Reversible) 절차를 자동 수행하는 기능

도 제공된다.

2020년엔 OpenQASM 지원을 위한 Scaffold-NISQ가 발표되었다[31]. 당초 Scaffold는 대규모 큐비트까지 수용 가능한 확장성 있는 모델로 제시되었는데 현실적으로 단기간, 중규모의 수요 및 IBM 클라우드 환경 맞춤으로 50~100큐비트 양자 소자를 목표로 하는 경량화 Scaffold-NISQ로 빠르고 쉬운 배포 버전이 공개된 것이다. NISQ Benchmark, IBM Backend Simulator 연동과 이에 따른 자원 계수기가 제공되며 특히 LLVM에 최적화 pass를 추가할 수 있어 유연성이 향상된다. OpenQASM용으로 컴파일러는 -b(basic), -f(flattened), -o(Optimized QASM), -T(Toffoli 분해), -R(Rotation 분해) 등의 선택지를 제공하며 OpenQASM에 맞추어 다차원을 단일 요소 배열로 분해하는 과정이 반영되어 있다. 이 사례를 통해서도 IBM 양자 클라우드의 영향력을 가늠할 수 있으며 어셈블리 언어는 OpenQASM이 사실 표준의 역할을 하고 있다.

2021년 4월에는 그림 3과 같은 컴파일과 실행 모델 기반의 OpenQASM 3이 발표되었는데 기본



출처 Reprinted from [32], CC-BY 4.0.

그림 3 OpenQASM 양자프로그램 컴파일과 실행 모델

게이트, 큐비트 매핑, 타이밍, 펄스 및 제어 흐름을 정의한 v2로부터 게이트 선언에서 제어 부분에 융통성을 향상하거나 서브루틴 정의 등을 확장·제공하며 하드웨어에 독립 혹은 종속적인 컴파일 단계를 지원하고 현재 30개에 달하는 컴파일러의 세부 변환 및 최적화 단계가 제공되고 있다[17,32].

Ⅳ. 양자컴퓨팅 검증 소프트웨어 동향

시뮬레이터는 알고리즘 개발에 중요한 역할을 하며 상태 공간 확장성에 관련이 크고 하위 수준에서 상위 논리 수준까지 다양한 동작 확인에 사용된다. 대상과 규모에 따라 슈퍼컴퓨터를 이용하거나 하위 집합에 대한 양자 시뮬레이터가 가능하며, 이를 위한 자원 추정은 매우 중요한 역할을 하므로 많은 플랫폼에서 필수적으로 제공 및 개선하고 있다[33]. 올해 MIT-하버드의 초저온 원자센터 연합팀은 루비듐 원자 기반으로 사상 최대 규모인 256큐비트 양자 시뮬레이터를 개발하여 기하급수적으로 더 많은 정보를 저장하고 처리할 수 있는 계기를 마련하였다[34]. 한편 D-Wave Systems는 양자 어닐링 방식으로 구글과 협업하여 2,000큐비트로 수십 년 된 문제를 고전컴퓨팅 대비 3백만 배 빠르게 모의시험을 진행하여 양자 계산의 장점을 다시 한번 확인하였다[35].

양자 프로그래밍은 복잡도나 측정 시 상태 붕괴하는 특성으로 검증과 디버깅이 난해하다. 따라서 양자컴퓨팅 도구의 검증(V&V: Validation & Verification)도 향후 많은 도전이 필요하고 발전이 필요한 영역이다[36]. 디버깅을 위한 assertion도 보조 변수의 측정을 이용하여 제한적으로 수행할 수 있으며 자원 추정기를 이용한 중단점(Breakpoint) 설정 전략이 가능하다. 양자 프로그래밍은 IBM Quan-

tum Composer나 ProjectQ, OpenQL QuantumInspire 등 플랫폼 내에서 이미 가시화 도구로 제공되고 있으며 양자의 상태를 복소수로 나타내는 블로흐 구와 연동하여 표현하기도 하고 회로 재구성의 단계를 보여주는 PyZX 도구는 t|ket〉에서 연동되어 있다. 양자 오류 정정은 NISQ에서 FTQC로 가는 과정에서 중요한 연구 주제이며 다양한 진행이 이루어지고 있고 이 자체로 별도의 정리가 필요한 중요 요소이다. 알고리즘이나 도구의 객관적 성능 비교를 위해서는 RevLib나 QASMBench 등 기존의 자료를 활용할 수 있다[37].

Intel은 2016년 qHipster라는 분산 고성능 시험 환경을 구축하였는데 2020년에는 HPC(High Performance Computing) 인프라를 이용하여 회전 각도 자동 변경 노이즈 게이트와 확률 회로 적용, 병렬 작동의 성능을 포함하여 최대 42큐비트까지 독립 혹은 백엔드로 동작할 수 있는 IQS(Intel Quantum Simulator)를 발표하였다[38].

프린스턴 대학교 연구진의 TriQ라는 도구 체인은 IBM 양자 클라우드에서 매일 제공하는 교정 정보를 반영하여 큐비트의 완화(Relaxation) 시간, 일관성(Coherence) 시간, 게이트 오류, 판독 오류 등 하드웨어의 실제 최신 정보를 반영한 컴파일러 최적화 알고리즘을 실행한다. 큐비트 할당에 있어서 선형 제약 조건을 입력으로 최적화하고 SMT(Satisfiability Modulo Theories) Solver를 적용하며 경험적 기법인 비중 노드/에지 우선 탐욕 기법(GreedyV/E)으로 기존 R-SMT 기법의 3시간 대비 1초 이하 소요의 컴파일 성능과 수백 큐비트 NISQ 시스템 적용성을 확인하였다[39].

2019년 시카고 및 컬럼비아 대학교와 함께 IBM 양자 클라우드 사용자의 저변 확대나 알고리즘 개선을 위해 CertiQ라는 검증 프레임워크도 진행하고 있으며 많은 단계를 자동화하였다[17]. 인터

표 3 유용한 양자 최적화 도구

도구	최적화
Feynman	x-rotation, CNOT-count, T-depth
Newsynth/ Gridsynth	1-qubit Z-rotations to Clifford+T
TOpt	Clifford+T to Clifford+T, T-gate minimization
IonQ's tool	Clifford + x-rotations + Toffoli to Clifford + x-rotations
RevKit	Look-up table hierarchical reversible synthesis (LHRS), CNOT minimization
pQCS	Multi-qubit unitary to Clifford+T
IBM QX mapping SU(4)	SU(4) to CNOT+SWAP+1-qubit gates

출처 Reproduced with permission from [40], CC-BY.

페이스에 따라 작성된 컴파일러 패스를 사용하여 Qiskit에서 실행 가능한 코드가 생성되고 양자 회로 등가성 확인용 계산이 가능하며 널리 사용되는 데이터 구조, 회로 변환 함수, 검증된 라이브러리가 제공된다.

2018년에는 양자 알고리즘과 오류 수정에 초점을 맞춘 연구자들이 모여 언어의 기본 구조와 명령 집합 모델, FTQC, 최적화, 자동 변환, 디버깅, 시험, 검증 도구, 응용, 추상화 등에 관하여 의미 있는 연구 발표 및 논의가 있었는데[40], 이 당시 Verification 지원은 QWIRE만 가능했지만 TriQ나 CertiQ의 발표에 따라 Scaffold나 Qiskit도 검증이 지원되는 언어로 분류할 수 있다고 판단된다. 이 논의 보고서 중 유용한 양자 최적화 도구들을 표 3으로 정리하였다.

V. 결론

양자컴퓨터는 생화학, 금융, 에너지, 자동차, 기후, 물류, 기계 학습, 인공지능 등 다양한 응용 분야에서 기존의 슈퍼컴퓨터로는 해결할 수 없는

문제에 적용되고 있다. 양자컴퓨터 하드웨어 시스템은 꾸준하게 연구 개발되고 있으며 2~3년 이내에 천 큐비트 이상의 양자 시스템이 개발되어 사용할 수 있을 것으로 예상된다. 하지만 큐비트가 갖는 짧은 결맞음 시간과 높은 오류율은 단시일 내에 해결될 수 없는 한계로 인식되고 있다.

양자컴퓨팅 소프트웨어는 다수의 기업과 대학교에서 양자 프로그래밍 언어, 양자 컴파일러, 양자 시뮬레이터 등 전체 소프트웨어 스택 구조로 연구 개발되고 있다. 양자 프로그래밍 언어는 다양한 양자 알고리즘을 보다 효과적으로 표현하기 위해 연구되고 있으며, 양자 컴파일러는 큐비트가 갖는 한계를 해결하고 극복할 방법 등에 관한 연구를 통해 양자 프로그램이 양자컴퓨터에서 효율적으로 실행될 수 있도록 한다. 본고에서 살펴본 주요 선도 플랫폼 소프트웨어 동향 분석을 통해 각 연구진이 중요하게 반영하고 있는 사항은 하드웨어 독립성, 프로그램의 확장성과 재사용성, 최적화를 위한 자원 추정과 분석에 집중하고 있음을 인식할 수 있다. 이외에 양자컴퓨팅 소프트웨어는 양자 오류 정정, 양자 실행 제어, 양자 암호, 양자통신 등 다양한 영역에서도 연구되고 있다.

양자컴퓨터는 현재 미국, 중국, 호주, 유럽 등지에서 기술 패권을 차지하기 위해 경쟁하고 있으며, 머지않은 미래에 상용화될 것으로 예상된다. 양자컴퓨터가 상용화 수준에 이르기 위해서는 다수의 안정적인 큐비트가 필요하지만, 이를 효율적으로 운용할 수 있는 소프트웨어 또한 매우 중요하다. 따라서 기존의 공개 코드 활용과 자율적인 커뮤니티 활동도 더욱 활성화되면서 양자컴퓨팅 소프트웨어를 구성하는 전체 스택에 대한 연구개발은 반드시 국가적 차원에서의 지속적인 추진을 기대한다.

용어해설

NISQ 잡음이 있는 중규모 양자컴퓨팅, 약 50에서 수백 큐비트를 포함하지만 내결함성에 도달할 만큼 충분히 진보되지 않았고, 양자 우위로부터 지속 가능한 이익을 얻을 만큼 충분히 크지 않은 양자 프로세서 수준을 이름

LLVM Low Level Virtual Machine은 컴파일러의 기반 구조로 컴파일, 링크, 실행 상황에서 언어에 상관없이 최적화를 쉽게 구현할 수 있도록 제공됨

Hoare Triple 프로그램의 정확성 추론을 위한 논리 형식 중 코드 실행이 계산 상태를 변경하는 방법을 설명함; 사전 조건 P, 일련의 프로그램 문장 S, 사후 조건 Q로 구성되며 {P}S{Q} 의미는 S가 실행되기 전에 P가 참이고, S의 실행이 종료되면, 사후에 Q가 참임을 나타냄

약어 정리

CNOT	Controlled NOT gate
CTQG	Classical–To–Quantum Gate (in ScaffCC)
FTQC	Fault–Tolerant Quantum Computing
LLVM	Low Level Virtual Machine
NISQ	Noisy Intermediate–Scale Quantum
QASM	Quantum Assembly Language
QCOR	QP language and a compiler(ORNL)
QEC	Quantum Error Correction
QISA	Quantum Instruction Set Architecture
ScaffCC	Scalable compilation and analysis framework
Scaffold	“Put things together” (Quantum PL)
XACC	eXtreme–scale Accelerator Programming Framework

참고문헌

[1] 전미 과학·공학·의학한림원, “양자컴퓨팅 발전과 전망,” 에이콘, 2021.

[2] E. Grumbling and M. Horowitz, “Quantum computing: Progress and prospects,” in National Academy of Sciences, Engineering, and Medicine, Washington, DC, USA, 2019, pp. 135-155.

[3] F.T. Chong et al., “Programming languages and compiler design

for realistic quantum hardware," Nature, vol. 549, 2017, pp. 180-187.

[4] 임승혁, "범용양자컴퓨터," KISTEP 기술동향브리프, 2019.

[5] 한상기, "양자컴퓨팅을 지원하는 클라우드서비스 현황," 씨앗 이슈리포트, 2020.

[6] 최병수, "양자컴퓨팅시스템 개발 및 활용 동향," 전자통신동향분석, 제31권 제2호, 2016, pp. 84-94.

[7] 박성수 외, "양자정보통신기술 현황과 전망," 전자통신동향분석, 제34권 제2호, 2019, pp. 60-72.

[8] 한상욱 외, "양자통신 및 양자컴퓨팅 분야 소개 및 연구동향," 융합연구리뷰, vol. 6, no. 3, 2020, pp. 31-58.

[9] 백충헌 외, "양자점 큐비트 기반 양자컴퓨팅의 국외 연구 동향분석," 전자통신동향분석, 제35권 제2호, 2020, pp. 79-88.

[10] 서화정 외, "양자컴퓨터와 양자 내성 암호 동향," ITFIND 주간기술 동향, 1979호, 2021, pp. 2-13.

[11] 이해웅, "양자 정보학 강의," 사이언스북스, 2017.

[12] 잭 히더리, "양자컴퓨팅: 이론에서 응용까지(Quantum computing: An applied approach)," 에이콘, 2020.

[13] 정지형, "양자정보기술 동향 및 시사점," ETRI Insight: Insight Report, 2019. 4.

[14] R. LaRose, "Overview and comparison of gate level quantum software platforms," Quantum, vol. 3, 2019, p. 130.

[15] Open-Source Quantum Software Projects, https://github.com/qosf/awesome-quantum-software

[16] R.S. Smith et al., "An open-source, industrial-strength optimizing compiler for quantum programs," Quantum Sci. Technol., vol. 5, no .4, 2020, p. 044001.

[17] Y. Shi et al., "CertiQ: Mostly-automated verification of a realistic quantum compiler," arXiv preprint, CoRR, 2020, arXiv: 1908.08963v5.

[18] D.S. Steiger et al., "ProjectQ: An open source software framework for quantum computing," Quantum, vol. 2, 2018.

[19] B. Bichsel et al., "Silq: A high-level quantum language with safe uncomputation and intuitive semantics," in Proc. ACM SIGPLAN Conf. Program. Lang. Des. Implementation, (London, UK), June 2020, pp. 286-300.

[20] A. JavadiAbhari et al., "ScaffCC: Scalable compilation and analysis of quantum programs," Parallel Comput., vol. 45, 2015, pp. 2-17.

[21] A.J. Abhari et al., "Scaffold: Quantum programming language," TR 934-12, Princeton University Nj Department of Computer Science, 2012.

[22] A.J. McCaskey et al., "XACC: A system-level software infrastructure for heterogeneous quantum-classical computing," Quantum Sci. Technol. vol. 5, no. 2, 2020, p. 024002.

[23] J.R. Reinders, "Intel C/C++ compilers complete adoption of LLVM," Aug. 9, 2021, https://software.intel.com/content/www/us/en/develop/blogs/adoption-of-llvm-complete-icx.html

[24] T.M. Mintz et al., "QCOR: A language extension specification for the heterogeneous quantum-classical model of computation," arXiv preprint, CoRR, 2019, arXiv:1909.02457.

[25] S. Sivarajah et al., "t|ket>: A retargetable compiler for NISQ devices," Quantum Sci. Technol., vol. 6, no. 1, 2020.

[26] N. Khammassi et al., "OpenQL: A portable quantum programming framework for quantum accelerators," arXiv preprint, CoRR, 2020, arXiv:2005.13283.

[27] K. Bertels et al., "Quantum computing—from NISQ to PISQ," IEEE Micro, vol. 41, no. 5, 2021.

[28] https://en.wikipedia.org/wiki/Quantum_programming

[29] D.A. Sofge, "A survey of quantum programming languages: History, methods, and tools," in Proc. Int. Conf. Quantum, Nano Micro Technol. (ICQNM 2008), (Sainte Luce, Martinique, France), Feb. 2008, pp. 66-71.

[30] A.J. Landahl et al., "Jaqal, the quantum assembly language for QSCOUT," arXiv preprint, CoRR, 2020, arXiv: 2003.09382.

[31] A. Litteken et al., "An updated LLVM-based quantum research compiler with further OpenQASM support," Quantum Sci. Technol. vol. 5, no. 3, 2020.

[32] A.W. Cross et al., "OpenQASM3: A broader and deeper quantum assembly language," arXiv preprint, CoRR, 2021, arXiv: 2104.14722v1.

[33] List of Quantum Simulators, https://quantiki.org/wiki/list-qc-simulators

[34] https://phys.org/news/2021-07-team-quantum-simulator-qubits-largest.html

[35] https://www.zdnet.com/article/a-quantum-computer-just-solved-a-decades-old-problem-three-million-times-faster-than-a-classical-computer/

[36] A. Miranskyy and L. Zhang, "On testing quantum programs," arXiv preprint, CoRR, 2018, arXiv: 1812.09261v1.

[37] QASMBench Benchmark Suite, https://github.com/pnnl/QASMBench

[38] G.G. Guerreschi et al., "Intel quantum simulator: A cloud-ready high-performance simulator of quantum circuits," arXiv preprint, CoRR, 2020, arXiv: 2001.10554v2.

[39] P. Murali et al., "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," arXiv preprint, CoRR, 2019, arXiv: 1909.02457.

[40] M. Mosca et al., "Report from dagstuhl seminar 18381: Quantum programming languages," Dagstuhl Reports, vol. 8, no. 9, 2018, pp. 112-114.